Prof. Mrudula Mukadam

BIG DATA TECHNOLOGY

Assignment 4

March 26, 2020
Ahmed Hamdy
611114

Q1. Write an in-mapper combiner algorithm for the “average problem”. (Pseudo code only; show reducer method too.)

Pseudo-code for the average problem with the in mapper combining:

```
class Mapper
        method setup ():
                memory ← new HashMap
        method map (String t, int r):
                if memory doesn’t have key t:
                        add <t, pair (r, 1)> to memory
                else:
                        pair (s, c) ← get value of t from memory
                        add <t, pair (s + r, c + 1)> to memory
        method cleanup ():
                for all <t, pair (s, c)> in memory do:
                        Emit (t, pair (s, c))
class Reducer
        method reduce (String t, pairs [(s1, c1), (s2, c2), …]):
                sum ← 0
                count ← 0
                for all pair (s, c) in pairs [(s1, c1), (s2, c2), …] do:
                        sum ← sum + s
                        count ← count + c
                average ← sum / count
                Emit (t, average)
```

Q2. Table question with part a, b, c

Assume that there are three reducers. Note that Reducer 1 runs on Machine1. Reducer 2 runs on Machine2. Reducer 3 runs on Machine3. Further, *let the partitioner assign all words starting from letter 'a-j' to Reducer 1, all words starting from letter 'k-q' to reducer 2 and everything else to Reducer 3*.

Also assume that there are six input splits as follows:

Input split1: [cherry mango olive cherry]
[plum cherry banana cherry]
Input split2: [cherry banana radish radish]
[carrot banana mango cherry]
Input split3: [banana kiwi plum banana]
[mango cherry kiwi banana]
Input split4: [apple mango carrot plum]
[radish kiwi banana olive]
Input split5: [olive banana radish kiwi]
[cherry kiwi olive cherry]
Input split6: [banana radish plum banana]
[olive cherry banana radish]

Input splits 1, 2 are on Machine 1, input splits 3, 4 are on Machine 2 and input splits 5, 6 are on Machine 3.

a) Illustrate the word count algorithm with no combiner, no in-mapper combining. *show mapper o/p, reducer i/p and reducer o/p*

b) Illustrate the word count algorithm with combiner, no in-mapper combining. *show mapper o/p, combiner o/p, reducer i/p and reducer o/p*

c) Illustrate the word count algorithm within mapper combiner. *show mapper o/p, reducer i/p and reducer o/p*

Remember to show the sorted mapper output that gets stored locally.

Note: Illustrate means show mapper o/p, combiner o/p (if using combiners), reducer i/p and reducer o/p.

a) No combiner, no in mapper combining

Machine 1		Machine 2		Machine 3	
Mapper 1 – Input Split 1- output		Mapper 3 – Input Split 3-output		Mapper 5 – Input Split 5-output	
<cherry, 1>	<plum, 1>	<banana, 1>	<mango, 1>	<olive, 1>	<cherry, 1>
<mango, 1>	<cherry, 1>	<kiwi, 1>	<cherry, 1>	<banana, 1>	<kiwi, 1>
<olive, 1>	<banana, 1>	<plum, 1>	<kiwi, 1>	<radish, 1>	<olive, 1>
<cherry, 1>	<cherry, 1>	<banana, 1>	<banana, 1>	<kiwi, 1>	<cherry, 1>
Mapper 1 – output file		Mapper 3 – output file		Mapper 5 – output file	
<banana, 1>		<banana, 1>		<banana, 1>	
<cherry, 1>		<banana, 1>		<cherry, 1>	
<cherry, 1>		<banana, 1>		<cherry, 1>	
<cherry, 1>		<cherry, 1>		<kiwi, 1>	
<cherry, 1>		<kiwi, 1>		<kiwi, 1>	
mango, 1>		<kiwi, 1>		<olive, 1>	
<olive, 1>		<mango, 1>		<olive, 1>	
<plum, 1>		<plum, 1>		<radish, 1>	
Mapper 2–Input Split 2 –output		Mapper 4 – Input Split 4-output		Mapper 6 – Input Split 6-output	
<cherry, 1>	<carrot, 1>	<apple, 1>	<radish, 1>	<banana, 1>	<olive, 1>
<banana, 1>	<banana, 1>	<mango, 1>	<kiwi, 1>	<radish, 1>	<cherry, 1>
<radish, 1>	<mango, 1>	<carrot, 1>	<banana, 1>	<plum, 1>	<banana, 1>
<radish, 1>	<cherry, 1>	<plum, 1>	<olive, 1>	<banana, 1>	<radish, 1>
Mapper 2 – output file		Mapper 4 – output file		Mapper 6 – output file	
<banana, 1>		<apple, 1>		<banana, 1>	
<banana, 1>		<banana, 1>		<banana, 1>	
<carrot, 1>		<carrot, 1>		<banana, 1>	
<cherry, 1>		<kiwi, 1>		<cherry, 1>	
<cherry, 1>		<mango, 1>		<olive, 1>	
<mango, 1>		<olive, 1>		<plum, 1>	
<radish, 1>		<plum, 1>		<radish, 1>	
<radish, 1>		<radish, 1>		<radish, 1>	
Shuffle & Sort					
Reducer 1 input		Reducer 2 input		Reducer 3 input	
<apple, [1]>		<kiwi, [1,1,1,1,1]>		<radish, [1,1,1,1,1,1]>	
<banana, [1,1,1,1,1,1,1,1,1,1,1]>		<mango, [1,1,1,1]>			
<carrot, [1,1]>		<olive, [1,1,1,1,1]>			
<cherry, [1,1,1,1,1,1,1,1,1,1]>		<plum, [1,1,1,1]>			

Reducer output is the same for all the cases:

Reducer 1 output	Reducer 2 output	Reducer 3 output
<apple, 1> <banana, 11> <carrot, 2> <cherry, 10>	<kiwi, 5> <mango, 4> <olive, 5> <plum, 4>	<radish, 6>

b) With combiner, no in mapper combining (assume that the combiner will work all the time)

Machine 1		Machine 2		Machine 3	
Mapper 1 – Input Split 1- output		Mapper 3 – Input Split 3- output		Mapper 5 – Input Split 5- output	
<cherry, 1>	<plum, 1>	<banana, 1>	<mango, 1>	<olive, 1>	<cherry, 1>
<mango, 1>	<cherry, 1>	<kiwi, 1>	<cherry, 1>	<banana, 1>	<kiwi, 1>
<olive, 1>	<banana, 1>	<plum, 1>	<kiwi, 1>	<radish, 1>	<olive, 1>
<cherry, 1>	<cherry, 1>	<banana, 1>	<banana, 1>	<kiwi, 1>	<cherry, 1>
Combiner 1 output– saved locally as mapper1 output file		Combiner 3 output– saved locally as mapper3 output file		Combiner 5 output– saved locally as mapper 5 output file	
<banana, 1>		<banana, 3>		<banana, 1>	
<cherry, 4>		<cherry, 1>		<cherry, 2>	
<mango, 1>		<kiwi, 2>		<kiwi, 2>	
<olive, 1>		<mango, 1>		<olive, 2>	
<plum, 1>		<plum, 1>		<radish, 1>	
Mapper 2–Input Split 2 – output		Mapper 4 – Input Split 4- output		Mapper 6 – Input Split 6- output	
<cherry, 1>	<carrot, 1>	<apple, 1>	<radish, 1>	<banana, 1>	<olive, 1>
<banana, 1>	<banana, 1>	<mango, 1>	<kiwi, 1>	<radish, 1>	<cherry, 1>
<radish, 1>	<mango, 1>	<carrot, 1>	<banana, 1>	<plum, 1>	<banana, 1>
<radish, 1>	<cherry, 1>	<plum, 1>	<olive, 1>	<banana, 1>	<radish, 1>
Combiner 2 output– saved locally as mapper2 output file		Combiner 4 output– saved locally as mapper4 output file		Combiner 6 output– saved locally as mapper6 output file	
<banana, 2>		<apple, 1>		<banana, 3>	
<carrot, 1>		<banana, 1>		<cherry, 1>	
<cherry, 2>		<carrot, 1>		<olive, 1>	

<mango, 1> <radish, 2>	<kiwi, 1> <mango, 1> <olive,1> <plum, 1> <radish, 1>	<plum, 1> <radish, 2>
Shuffle & Sort		
Reducer 1 input	Reducer 2 input	Reducer 3 input
<apple, [1]> <banana, [1,2,3,1,1,3]> <carrot, [1,1]> <cherry, [4,2,1,2,1]>	<kiwi, [2,1,2]> <mango, [1,1,1,1]> <olive, [1,1,2,1]> <plum, [1,1,1,1]>	<radish, [1,2,1,2]>

c) With in-mapper combining

Machine 1	Machine 2	Machine 3
Mapper 1 – Input Split 1- output file	Mapper 3 – Input Split 3- output file	Mapper 5 – Input Split 5- output file
<banana, 1> <cherry, 4> <mango, 1> <olive, 1> <plum, 1>	<banana, 3> <cherry, 1> <kiwi, 2> <mango, 1> <plum, 1>	<banana, 1> <cherry, 2> <kiwi, 2> <olive, 2> <radish, 1>
Mapper 2–Input Split 2 – output file	Mapper 4 – Input Split 4- output file	Mapper 6 – Input Split 6- output file
<banana, 2> <carrot, 1> <cherry, 2> <mango, 1> <radish, 2>	<apple, 1> <banana, 1> <carrot, 1> <kiwi, 1> <mango, 1> <olive,1> <plum, 1> <radish, 1>	<banana, 3> <cherry, 1> <olive, 1> <plum, 1> <radish, 2>
Shuffle & Sort		
Reducer 1 input	Reducer 2 input	Reducer 3 input
<apple, [1]> <banana, [1,2,3,1,1,3]> <carrot, [1,1]> <cherry, [4,2,1,2,1]>	<kiwi, [2,1,2]> <mango, [1,1,1,1]> <olive, [1,1,2,1]> <plum, [1,1,1,1]>	<radish, [1,2,1,2]>